On the Evaluation of NLP-based Models for Software Engineering

Izadi, Maliheh ; Ahmadabadi, Martin Nili

DOI
10.1145/3528588.3528665

Publication date
2022

Document Version
Final published version

Published in
Proceedings of the 2022 IEEE/ACM 1st International Workshop on Natural Language-Based Software Engineering (NLBSE)

On the Evaluation of NLP-based Models for Software Engineering

Maliheh Izadi
m.izadi@tudelft.nl
Delft University of Technology
Delft, Netherlands

Matin Nili Ahmadabadi
matin_nili@alumni.ut.ac.ir
University of Tehran
Tehran, Iran

ABSTRACT

NLP-based models have been increasingly incorporated to address SE problems. These models are either employed in the SE domain with little to no change, or they are greatly tailored to source code and its unique characteristics. Many of these approaches are considered to be outperforming or complementing existing solutions. However, an important question arises here: *Are these models evaluated fairly and consistently in the SE community?*. To answer this question, we reviewed how NLP-based models for SE problems are being evaluated by researchers. The findings indicate that currently there is no consistent and widely-accepted protocol for the evaluation of these models. While different aspects of the same task are being assessed in different studies, metrics are defined based on custom choices, rather than a system, and finally, answers are collected and interpreted case by case. Consequently, there is a dire need to provide a methodological way of evaluating NLP-based models to have a consistent assessment and preserve the possibility of fair and efficient comparison.

KEYWORDS

Evaluation, Natural Language Processing, Software Engineering

ACM Reference Format:
Maliheh Izadi and Matin Nili Ahmadabadi. 2022. On the Evaluation of NLP-based Models for Software Engineering. In *The 1st Intl. Workshop on Natural Language-based Software Engineering (NLBSE'22), May 21, 2022, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3528588.3528665

1 INTRODUCTION

Researchers have been using NLP-models to solve a diverse set of SE problems such as code generation, completion, summarization, bug fixing, question answering, test case generation, documentation, and many more. As these models attract more researchers and the number and diversity of studies grows, it is imperative to have good evaluation measures and techniques to assess them properly. These measures should be consistent throughout the literature in order to conduct fair and comparable comparisons. To understand the evaluation of NLP models, we reviewed the field in the past five years and report the results here. To the best of our knowledge, we are the first to conduct a systematic literature review on evaluation of NLP-based models to understand the underlying patterns, identify the challenges, and recommend future research direction.

2 METHODOLOGY

We conducted our systematic review using the following protocol. Our main research question is "How are NLP-based models evaluated in SE?". Search phrases in the title, abstract or body of a paper are *NLP*, *natural language processing*, *code*, and *evaluation*. Papers must be peer-reviewed, written in English, and be published after 2017 by one of the following SE prominent conferences and journals: *ICSE*, *ESEC/FSE*, *ASE*, *IEEE TSE*, *ACM TOSEM*, and *EMSE*. We used *Google Scholar* as the source, and retrieved 157 papers. Two of the authors manually inspected all papers to identify the papers that propose an NLP-based model to solve a SE problem. Finally, 53 papers were excluded because of one or more of the following reasons: the paper's scope was unrelated to NLP and SE, the main proposed model was not based on NLP, or it was a secondary or duplicate study. Next, we present the result of the review on the remaining 104 included papers. More information on the protocol and papers can be found in our GitHub repository.[1]

3 EVALUATION OF NLP-BASED MODELS

There are two approaches to evaluation *intrinsic* with a focus on intermediary goals (sub-tasks), and *extrinsic* for assessing the performance of the final goal. NLP-based models in SE are generally evaluated with one or more of the following metrics.
(1) **Automatically**: Automatic evaluation consists of three groups, namely (i) metrics for assessing the results of classification models such as *Accuracy*, *Precision*, *Recall*, and *F* measure, (ii) metrics for assessing recommendation lists including *Top@n* or ranked versions such as *MRR* and *MAP*, and (iii) metrics for analyzing the quality of generated text or source code including *BLEU*, *METEOR*, *ROUGE*, *CIDEr*, *chrF*, *Perplexity*, and *Levenshtein similarity* metrics.
(2) **Manually**: Manual assessment is more subjective and heeds the judgment of human participants. Researchers first select the relevant metric(s) to evaluate different aspects of the proposed model's output. Then, they invite a group of experts to assess the results based on the selected metrics. For instance, for a code summarization task, researchers use *informativeness* as an indicator of the quality of the generated summaries from the developers' perspective.

Automatic evaluation is easier, faster, and completely objective compared to the manual version. Thus most researchers opt to use automatic evaluation for assessing their models. However, human-based assessments can potentially convey more information for several aspects of a model, hence, they can be used to complement

NLBSE'22, May 21, 2022, Pittsburgh, PA, USA
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9343-0/22/05.
https://doi.org/10.1145/3528588.3528665

[1]https://github.com/MalihehIzadi/nlp4se_eval

automatic evaluation. Recently, Roy et al. [12] conducted an empirical study on the applicability and interpretation of automatic metrics for evaluation of the. code summarization task. With the help of 226 human annotators, they assessed the degree to which automatic metrics reflect human evaluation. They claim that less than 2 points improvements for an automatic metric such as BLEU do not guarantee systematic improvements in summarization quality. This makes the role of human assessment salient.

Although automatic measures are uniformly defined in the literature, manual metrics are harder to define, interpret and use. These measures must be properly indicative of a model's goal and performance. Furthermore, their definitions and usage must be kept consistent to have comparable results. Hence, in the following we review the most popular existing manual assessment measures in the SE domain and leave the rest of them (such as effectiveness, comprehensibility, time-saving, relatedness, rightness, usability, recency, grammatically correctness, advantageousness, diversity, self-explanatory, theme identification, and more) for a more comprehensive study. **Usefulness**: Several studies define usefulness as how useful participants find the proposed solution for solving the problem at hand [2, 4, 7, 11, 16]. Others define usefulness as the tendency or preference of users to use their proposed model [17]. Jiang et al. [8] assess the usefulness of its results based on both its accuracy and the difficulty of generating outputs. That is, they focus on how often the model works when it is indeed needed. **Naturalness**, **Expressiveness**, **Readability**, and **Understandability**: Roy et al. [13] define naturalness as how easy it is to read and understand generated outputs. They also use readability to measure to what extent the output is perceived as readable and understandable by the participants. Aghamohammadi et al. [1] define naturalness as how smooth, human-readable, and syntactically-correct are their outputs. Gao et al. [5] measure naturalness as the grammatical correctness and fluency of a generated sentence. Zhou et al. [18] use expressiveness as whether their model's output is clear and understandable. **Correctness** or **Content**: Huang et al. [6] define correctness as whether participants can find the correct API using their proposed tool, while Chen et al. [3] define it as a measure to verify the general correctness of the abbreviations and synonyms in their thesaurus. In Roy et al.'s [13] study, content means whether a summary correctly reflects the content of a test case. **Completeness** and **Informativeness**: Uddin et al. [14] define completeness as a complete yet presentable summarization of API reviews. Aghamohammadi et al. [1] define informativeness as how much of the important parts of a piece of code are covered by a generated summary. **Conciseness**: In Roy et al.'s [13] study, concise summaries do not include extraneous or irrelevant information. Zhou et al.[18] quantifies conciseness through answering whether the repair recommendation is free of other constraint-irrelevant information. **Relevance** or **Similarity**: Several studies define relevance as to how relevant is the model's output to the reference text or code [2, 5, 11, 16]. Others asked developers to rate the similarity, relatedness, and contextual or semantic similarity between outputs and reference texts [9, 10, 15].

4 DISCUSSION, AND FUTURE DIRECTION

We reviewed 104 studies to understand how NLP-based models are usually evaluated in the SE domain and provided the list of most used metrics. Next, we provide the main challenges for the evaluation of NLP-based models. (1) Both automatic and manual approaches can be utilized to provide a more holistic view of the performance of an NLP-based model, however, not all of the eligible studies use both of these approaches. (2) For the manual form of assessment, there exist numerous and sometimes conflicting or ambiguous evaluation metrics. This problem exacerbates in the case of measures with multiple definitions (e.g., informativeness) or in case of multiple metrics which are overlapping (e.g., naturalness, readability, understandability, and expressiveness). Some researchers evaluate different aspects of their model, (e.g., completeness, naturalness, or correctness) while others only address one or two aspects. As there are various aspects to each model, there should be a methodological way to first identify the most important aspects of a given SE task and then properly evaluate those aspects with concretely defined metrics. (3) In addition to the definition and use of the metrics, there is no standard for defining the set of answers. That is, some use yes/no answers, while others use *n*-point Likert scale or even free-format text answers. (4) Finally, identifying novel evaluation metrics or techniques can help SE researchers assess these models more thoroughly and where it matters. For example, for the automatic code completion task, predicting an identifier is more valuable and difficult than predicting a keyword. Hence, evaluating models for predicting any token is not very helpful. Through reviewing the literature, we took the first step toward addressing the challenges of proper evaluation for NLP-based models. We suggest future research focus on providing a systematic and consistent framework for evaluation of these models to (1) clearly define measures, (2) distinguish between different needs of SE tasks, and (3) determine the proper use of a measure in the context. Hopefully, a systematic way of evaluation makes it possible to conduct fair and correct evaluations for the NLP-based models in the SE field.

REFERENCES

[1] Alireza Aghamohammadi, Maliheh Izadi, and Abbas Heydarnoori. 2020. Generating summaries for methods of event-driven programs: An Android case study. *Journal of Systems and Software* 170 (2020), 110800.

[2] Liang Cai, Haoye Wang, Bowen Xu, Qiao Huang, Xin Xia, David Lo, and Zhenchang Xing. 2019. AnswerBot: an answer summary generation tool based on stack overflow. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1134–1138.

[3] Chunyang Chen, Zhenchang Xing, and Ximing Wang. 2017. Unsupervised software-specific morphological forms inference from informal discussions. In *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. IEEE, 450–461.

[4] Andrea Di Sorbo, Sebastiano Panichella, Carol V Alexandru, Corrado A Visaggio, and Gerardo Canfora. 2017. SURF: summarizer of user reviews feedback. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. IEEE, 55–58.

[5] Zhipeng Gao, Xin Xia, John Grundy, David Lo, and Yuan-Fang Li. 2020. Generating question titles for stack overflow from mined code snippets. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 29, 4 (2020), 1–37.

[6] Qiao Huang, Xin Xia, Zhenchang Xing, David Lo, and Xinyu Wang. 2018. API method recommendation without worrying about the task-API knowledge gap. In *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 293–304.

[7] Maliheh Izadi, Abbas Heydarnoori, and Georgios Gousios. 2021. Topic recommendation for software repositories using multi-label classification algorithms.

Empirical Software Engineering 26, 5 (2021), 1–33.

[8] Lin Jiang, Hui Liu, and He Jiang. 2019. Machine learning based recommendation of method names: how far are we. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 602–614.

[9] Siyuan Jiang, Ameer Armaly, and Collin McMillan. 2017. Automatically generating commit messages from diffs using neural machine translation. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 135–146.

[10] Zhongxin Liu, Xin Xia, Christoph Treude, David Lo, and Shanping Li. 2019. Automatic generation of pull request descriptions. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 176–188.

[11] Xiaoxue Ren, Xinyuan Ye, Zhenchang Xing, Xin Xia, Xiwei Xu, Liming Zhu, and Jianling Sun. 2020. API-misuse detection driven by fine-grained API-constraint knowledge graph. In *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 461–472.

[12] Devjeet Roy, Sarah Fakhoury, and Venera Arnaoudova. 2021. Reassessing automatic evaluation metrics for code summarization tasks. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1105–1116.

[13] Devjeet Roy, Ziyi Zhang, Maggie Ma, Venera Arnaoudova, Annibale Panichella, Sebastiano Panichella, Danielle Gonzalez, and Mehdi Mirakhorli. 2020. DeepTC-Enhancer: Improving the readability of automatically generated tests. In *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 287–298.

[14] Gias Uddin and Foutse Khomh. 2017. Automatic summarization of API reviews. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 159–170.

[15] Yaza Wainakh, Moiz Rauf, and Michael Pradel. 2021. IdBench: Evaluating Semantic Representations of Identifier Names in Source Code. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 562–573.

[16] Haoye Wang, Xin Xia, David Lo, John Grundy, and Xinyu Wang. 2021. Automatic Solution Summarization for Crash Bugs. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 1286–1297.

[17] Yu Zhao, Tingting Yu, Ting Su, Yang Liu, Wei Zheng, Jingzhi Zhang, and William GJ Halfond. 2019. Recdroid: automatically reproducing android application crashes from bug reports. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 128–139.

[18] Yu Zhou, Xin Yan, Taolue Chen, Sebastiano Panichella, and Harald Gall. 2019. DRONE: a tool to detect and repair directive defects in Java APIs documentation. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. IEEE, 115–118.